اصول پردازش تصویر

نیم‌سال اول ۱۳۹۹-۱۴۰۰

مدرس: دکتر مصطفی کمالی تبریزی

دانشکده علوم ریاضی

تمرین سری دوم ــ سوال دوم

نام و نام‌خانوادگی: سیدعلیرضا خادم

شماره دانشجویی: ۹۷۱۰۰۳۹۸

موارد لازم.

برای اجرا لازم است تا تصاویر greek_ship.jpg و patch.png در مسیر /EX2_Q2/images قرار داشته باشد.

روند کلی حل.

در ابتدا سعی شد حل این سوال با روش SSD انجام شود اما با توجه به اینکه نتیجه نتیجه خوبی با این روش به دست نمی‌آمد سراغ روش normalized cross correlation رفتیم. در این روش نتیجه بهتری نسبت به روش‌های cross correlation ، zero mean cross correlation و SSD حاصل می‌شد اما زمان اجرا در روش NCC نسبت به روش‌های دیگر بیشتر بود، در نتیجه همان طور که در توضیح تابع هم normalized_cross_correlation آمده است به جای $\overline{f_{m,n}}$ در رابطه‌یِ محاسبه NCC ، میانگین کلِ عکس را قرار دادیم تا محاسبه عبارت در زمان خطی انجام شود و نتیجه هم خوب بود. نتیجه‌هایی که می‌شد گرفت تا حد خوبی بهتر شده بود اما به دلیل اینکه در patch علاوه بر ستون، قسمتی از آسمان و دریا هم وجود داشت نتیجه‌ای که در تصویر 000.jpg در مسیر /EX2_Q2/images مشاهده می‌کنید به دست می‌آمد؛ در نتیجه تصمیم بر این شد تا با یک روشی patch را کوچکتر کنیم تا قسمت‌هایی اضافی حذف بشه. با انجام این کار با روش‌ای که در تابع cut_object_of_patch توضیح داده شده به 001.jpg در مسیر /EX2_Q2/images رسیدیم. در گام بعدی برای این که نتیجه بهتری به دست بیاید نتیجه Template Matching کانال‌های رنگی مختلف از patch و تصویر اصلی را مقایسه کردیم. تصویر 003.jpg ، 004.jpg و 005.jpg در مسیر /EX2_Q2/images به ترتیب مربوط به کانال‌های آبی، سبز و قرمز هستند. همانطور که مشاهده می‌کنید کانال آبی نتیجه بهتری میتواند بدهد، در نتیجه کانال آبی به عنوان مبنا در نظر گرفته شد. در نهایت هم یک Transformation Gamma روی نتیجه نهایی اعمال شد تا اون قسمت‌هایی که کاندیدهای اصلی بودند روشن تر شوند و در عوض اون قسمت‌هایی که کاندیدا نبودن تیره بشوند و تصویر gamma_trans_applyed.jpg در مسیر /EX2_Q2/images به دست آمد.

در این کد تعداد object هایی که در تصویر میخواهیم دیتکت بشوند را در متغیر number_of_object در فایل utilities.py باید مشخص کنیم. برای مثال در این سوال تعداد این object ها ۹ عدد است که شامل ۴ ستون در آب و ۵ ستون روی کشتی می‌شود.

در آخر هم باید به این نکته اشاره کنم که برای حل این سوال به این نکته توجه شده بود که ممکن است سایز

object هایی که در تصویر هستند و قرار است دیتکت شوند با سایز template متفاوت باشد که در این صورت یک راهکاری که وجود دارد سرچ کردن template با سایزهای متفاوت است. در این سوال در ابتدا پیاده سازی این موضوع انجام شده بود اما با توجه به اینکه در این سوال استثنائا بدون این کار هم نتیجه مطلوبی حاصل شد. آن بخش از پیاده‌سازی حذف گردید.

توضیح کد.

برنامه در مجموع حاوی ۲ فایل با فرمت py. می‌باشد که توضیحات هر فایل در پایین آمده است.

○ utilities.py

canny(src_image)

این تابع تصویر src_image را به عنوان ورودی می‌گیرد و متد تشخیص مرزِ canny را با high_thresh و low_thresh ای که در تابع محاسبه شده است روی آن اعمال می‌کند و تصویری که حاصل می‌شود را به عنوان خروجی برمی‌گرداند.

cut_object_of_patch(gray_template_patch, template_patch)

هدف از پیاده سازی این تابع این بوده است که template ای که در سوال داده شده بود علاوه بر ستون (که object اصلی ای بود که باید تشخیص داده می‌شد) قسمتی از آسمان و دریا هم در template و جود داشت و با این باعث می‌شد که نتیجه خوبی به حاصل نشه. این تابع gray_template_patch که نسخه سیاه و سفید template داده شده در سوال است و template_patch که همان template داده شده در سوال است را به عنوان ورودی می‌گیرد و با استفاده از تابع canny() مرز های gray_template_patch را همان طور که در تصویر template_edge_detected.jpg در مسیر EX2_Q2/images/ قابل مشاهده است به دست می‌آورد. سپس کوچکترین مستطیلی که شامل مرزهای تشخیص داده شده است را محاسبه می‌کند و آن قسمت از تصویرِ template_patch را (منظور از قسمت همان مستطیل‌ای است که محاسبه شده) همان طور که در تصویر new_patch.png در مسیر EX2_Q2/images/ قابل مشاهده است به عنوان خروجی برمی‌گرداند.

normalized_cross_correlation(src_image, input_template)

در این تابع پیاده‌سازی روش normalized cross correlation برای Template Matching انجام شده، با این تفاوت که در عبارتی که برای normalized cross correlation در اسلایدهای درس آمده و در پایین هم تصویر عبارت قرار داده شده است، به جای $\overline{f_{m,n}}$ ، میانگین کلِ عکس را قرار دادیم تا محاسبه عبارت در زمان خطی انجام شود (باید در نظر داشت که این کار مشروط به اینکه در ریزالت نهایی تاثیر منفی نداشته باشه انجام شده). درنهایت هم result اسکیل شده به ۰ تا

۲۵۵ و به عنوان خروجی باز گردانده شده است.

$$h[m,n] = \frac{\sum_{k,l}(g[k,l] - \bar{g})\left(f[m+k,n+l] - \overline{f_{m,n}}\right)}{\left(\sum_{k,l}(g[k,l] - \bar{g})^2 \sum_{k,l}\left(f[m+k,n+l] - \overline{f_{m,n}}\right)^2\right)^{0.5}}$$

mean template

mean image patch

draw_rectangle_aroung_detected_template(src_image, result, w, h , k)

این تابع src_image را به عنوان عکسی که قرار است در آن دور object های تشخیص داده شده مستطیل رسم شود ، result را به عنوان نتیجه normalized cross correlation ، w و h را به عنوان ابعاد template و k را به عنوان تعداد اشیایی که در عکس وجود دارد ورودی می‌گیرد و for ای می‌زند که تعداد اجرای آن k بار است و در هر بار اجرا نگاه می‌کند ماکزیمم result کجا قرار دارد، به مرکز آن نقطه یک مستطیل رسم می‌کند و یک ناحیه‌ای اطراف آن را مقادیرشان را مقدار کمی قرار می‌هد تا در اجراهای بعدی حلقه دیگر در آن قسمت‌ها object ای دیتکت نشود.

apply_gamma_transformation(src_image, gamma)

این تابع یک تصویر را به عنوان ورودی می‌گیرد و یک Gamma Transformation روی آن اعمال می‌کند و نتیجه اعمال را به عنوان خروجی برمی‌گرداند.

○ q۲.py

در این فایل ابتدا تصاویر greek_ship.jpg و patch.jpg را از مسیر /EX2_Q2/images لود شده است. بعد برای جلوگیری از overflow شدن مقادیر قرمت src_image و template به np.float_ تغییر داده شده است. در ادامه با توجه به توضیاحتی که در **روند کلی حل.** و **توضیح کد.** داده شد result محاسبه می‌شود و تحت عنوان res03.jpg در مسیر /EX2_Q2/results ذخیره می‌شود.